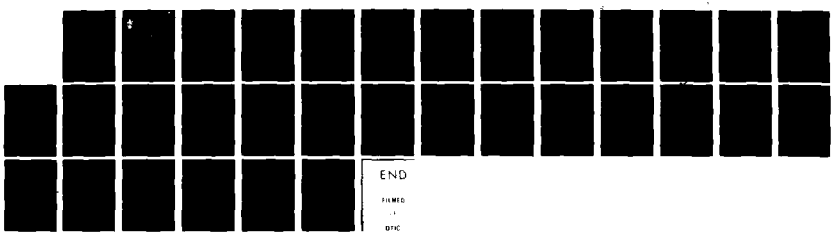
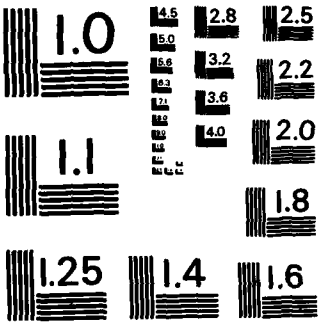
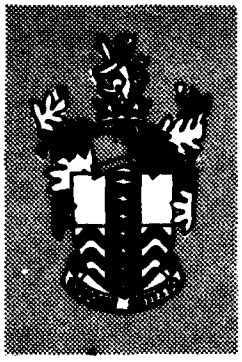
AD-A121 377 A TOOL FOR ANALYSING THE DESIGN AND PERFORMANCE OF A 1/1
MMASCOT ACP NETWORK -.. (U) ROYAL SIGNALS AND RADAR
ESTABLISHMENT MALVERN (ENGLAND) A C TYLER JUN 82
UNCLASSIFIED RSRE-MEMO-3475 DRIC-BR-85268 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A



RSRE
MEMORANDUM No. 3475

ROYAL SIGNALS & RADAR ESTABLISHMENT

A TOOL FOR ANALYSING THE DESIGN AND PERFORMANCE OF A MASCOT ACP NETWORK – INTRODUCTORY REPORT

Author: A C F Tyler

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

RSRE MEMORANDUM No. 3475

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3475

Title: A TOOL FOR ANALYSING THE DESIGN AND PERFORMANCE OF A MASCOT ACP NETWORK - INTRODUCTORY REPORT

Author: A C F Tyler

Date: June 1982

SUMMARY

This paper describes a tool to facilitate analysis of MASCOT systems. Both topological and behavioural analysis of an ACP network can be performed using techniques of graph theory and event simulation. The simulation environment comprises a number of simulating tasks running under a general purpose operating system that supports pseudo parallelism between the tasks. A database is created to hold the ACP diagram including node attributes. The topological input is derived from either a set of MASCOT construction commands or from a graphics terminal. Node attributes used in the simulation are created from a set of language statements describing in skeleton form the nature of MASCOT activities, IDAs and devices. Performance data is recorded using a data monitoring program.

The paper proposes that existing MASCOT machines be instrumented to interface to the simulation program and the data monitor. For use where this is not possible (and to make the tool more general purpose) a method of characterising MASCOT machines from language statements and interpreting these characterisations is presented.

RSRE MEMORANDUM NO 3475

A TOOL FOR ANALYSING THE DESIGN AND PERFORMANCE OF A MASCOT ACP NETWORK

A C F Tyler

CONTENTS

1 GENERAL INTRODUCTION

Since the early days of Phillips diagrams it has been recognised that the decomposition of a real-time system into data components and processes is largely an intuitive matter. The advent of the MASCOT formalism has not changed matters. Experience with the MASCOT ACP diagram as a design tool has however, highlighted some pitfalls of its own. For example, it is very tempting for the new MASCOT convert to become so enthusiastic about the modular approach that activities and IDA's proliferate unnecessarily. Often, perhaps as a reaction against the monolithic program approach, the function assigned to a MASCOT activity is too trivial. This is only one of the considerations incumbent upon the system designer. How much data to put in a single IDA and whether to use a channel or a pool are others. We might think of these as the topological design issues (ie how we draw the ACP network). Other issues which here we will call behavioural issues include data throughput rates, identification of

bottlenecks and deadly embrace situations as well as consideration of processing power requirements. The latter may involve partitioning an ACP diagram for implementation on several processors, stores and data highways. Traditionally, behavioural issues have dominated the design of the software. The notion of software prototyping implies that consideration of the eventual computing hardware configuration should be decoupled totally from the design phase and the use of the MASCOT ACP diagram as a design aid makes this possible. The premise that the software structure should determine the hardware configuration and not vice versa appears to be valid. However, the fact remains that the ideal hardware never exists and often (particularly in embedded systems) the problem of mapping the software structure on to limited hardware is not trivial.

This paper describes an analysis tool to help investigate both topological and behavioural aspects of an ACP network. The analyser is to be implemented on a minicomputer running with an operating system.

2 AIMS

The purpose of the analysis tool is to investigate in a controlled manner the nature of a MASCOT network and the MASCOT machine which supports it. By this means evidence should be produced that will show the nature of the performance achieved by the network and separate overheads which are inherent in the MASCOT technique from those that depend on the topology of the network. A tool that enables performance measurements to be made on a given application ACP network design will enhance the current set of software development facilities available to the MASCOT designer and engineer. At least two uses for the tool are identified. Firstly, the ACP network design team may use it to demonstrate the feasibility of the initial design ideas by modelling the system in a skeleton form and observing the general effects of certain allocations of functions to software modules. For example, it might be possible to assess the optimum size for an activity with a given set of data access requirements. Secondly, and later in the development process, more detailed knowledge of the hardware to be used and the function of each MASCOT module would become available. In this case the model would be a useful way of giving confidence that the ACP network could in fact meet the required performance constraints. A requirement of the tool is that it is interactive and useable by a software engineer at a low cost computer terminal with a graphics capability.

3 THE MAIN ELEMENTS OF THE TOOL

The tool comprises five main elements or subsystems as shown in figure 3.1.

3.1 The Topological Analysis Subsystem

Two types of topological analysis are possible with this subsystem. The first is to check that the MASCOT rules for interconnection of modules are obeyed. This validation process will be an essential pre-requisite for any further analysis of the ACP network.

The second type of analysis is to find all data paths through the ACP network (say, from one particular node to any other selected node). In this way data loops can be discovered which might not have been planned and which might give rise to deadly embrace situations, depending on activity synchronisation protocols (ie access procedure design). Chapter 4 describes further this type of analysis.

3.2 The Behavioural Analysis Subsystem

The purpose of this subsystem is to provide an environment in which to simulate the ACP network and the underlying MASCOT machine(s). The overall aim is to assess the ability of the system to cope with the data flow rates from/to data sources/sinks and if possible to identify bottlenecks. The data to be calculated for each MASCOT module is as follows:-

1 For each activity:-

   a processor time consumed (as a percentage of the total available processor capacity);

   b time spent waiting for data;

   c time spent pending (caused by mutual exclusion).

2 For each IDA:-

   a average queue length/contention rate for each mutual exclusion variable (control queue);

   b average buffer occupancy;

   c buffer-full time;

   d buffer-empty time.

3 For each device:-

   Whether crisis times are met.

The simulated environment is provided by an event coordinator and data is recorded using a program called the ACP monitor. Two types of simulation entity exist. The first type is an instrumented MASCOT system in which it is assumed that all activities are under the control of a single MASCOT machine which runs on a single processor. The second type is a device simulator and represents a single uni-directional device. In simulating a typical MASCOT network there will be one or several occurrences of these types of entity according to the topology of the network. The event coordinator, ACP monitor and the simulation entities are from now on called tasks, referred to as the event coordinator task, the ACP monitor task and simulating tasks respectively. All tasks run under the control of a supporting operating system. The simulating tasks run in conjunction with the event coordinator which controls simulated time. Chapters 5, 6, 7 and 8 give more detail for the event coordinator, the ACP monitor, device simulators and instrumented MASCOT systems respectively. Chapter 10 describes how device simulators and instrumented MASCOT systems are built from a set of characterisations (descriptions) of the elements of a MASCOT system and the devices with which it interacts.

3.3 The Database Subsystem

This subsystem comprises the MASCOT system database, a database updating program and database update records. The MASCOT system database contains a representation of the connectivity of the MASCOT ACP network together with a description of its node- and interconnection- attributes.

The nodes are the MASCOT modules (activities, channels and pools) and all devices. The interconnections are data paths which might represent communications media such as buses and data highways. Examples of node attributes are:-

node types,
node names,
node numbers,
IDA data access times,
a description of the MASCOT machine with which the node is associated (for an activity: the MASCOT machine that schedules it, for an IDA: the MASCOT machine on which the access procedures execute),
attributes that describe the hardware onto which a node maps (for an activity: on what processor it executes, for an IDA: on what processor its access procedures execute, for a device: a description of the function of the device).

Interconnection attributes define the speed of data transmission between nodes characterised by use of a selected communications medium. The hardware configuration is described by a number of environment attributes which specify the character of the MASCOT machine(s) in terms such as the length of time to execute MASCOT primitives and scheduling policy or policies to be used. The hardware attributes define, for example the speed of processor used and the characteristics of data communication media used. Chapter 9 describes in more detail the design of the database.

The database updating program (see figure 3.1) reads a set of update records that specify one of the following actions:-

i Add a new attribute;

ii Delete an existing attribute;

iii Update an existing attribute;

iv List attributes;

v Attach attributes;

vi Detach attributes.

Attributes are related to the ACP nodes held in the MASCOT system database. They originate from one of two sources:-

1 the user at a graphics terminal drawing ACP diagrams;

2 statements in a command language read either from the terminal of from a filestore.

Chapter 9 describes attribute management more fully.

3.4 The Graphics Subsystem

Software is provided to drive an interactive graphics terminal and produce a picture of the ACP diagram. This is shown as the graphics driver in Figure 3.1. The terminal user has the ability to draw and alter ACP diagrams with the following aids:-

i A set of building elements to enable an ACP diagram to be drawn on the screen. These elements are: activity symobols, IDA symbols, device symbols, straight lines (for node connection) and characters (for labelling).

ii A command decoder to respond to requests for element-selection, connection, labelling, picture-layout and picture-selection.

3.5 The Control Subsystem

This subsystem controls the operation of the tool. A command interpreter processes all terminal and filestore input including graphics commands. It converts the user input format into database update records and graphics commands. It converts the user input format into database update records and graphics control commands.

4 TOPOLOGICAL ANALYSIS METHODS

4.1 Introduction

Topological analysis is performed on a representation of the ACP network. When a network is created the adjacency matrix is formed which defines its connectivity. The elements of the matrix are labels that are strings formed by concatenating the digits of the node numbers of the connected nodes. For example, the element a(5,11) representing a connection from node number 5 to node number 11 has the label 05-11.

4.2 MASCOT Rules Validation

The rules governing connection of MASCOT modules (nodes) are:-

i An activity may only be connected to IDAs;

ii An IDA may only be connected to activities or devices.

4.3 Path Finding

The path finding algorithm produces a set of concatenations of the elements of the adjacency matrix (ie the arc labels) which define all the paths through the network which only pass once through any node (called elementary paths).

The technique used is described by Carre (reference (1)) it involves the calculation of the weak closure of the adjacency matrix with respect to binary operations defined for an algebra called a path algebra. The closure is calculated as the least solution of a matrix equation using a method analogous to the Jordan method for matrix inversion used in classical linear algebra..

The algorithm is run in response to a command from the user. The above technique yields a matrix whose elements give a description of network paths in terms of the node-connection labels. This information is relayed to an output device which may be the graphics output.

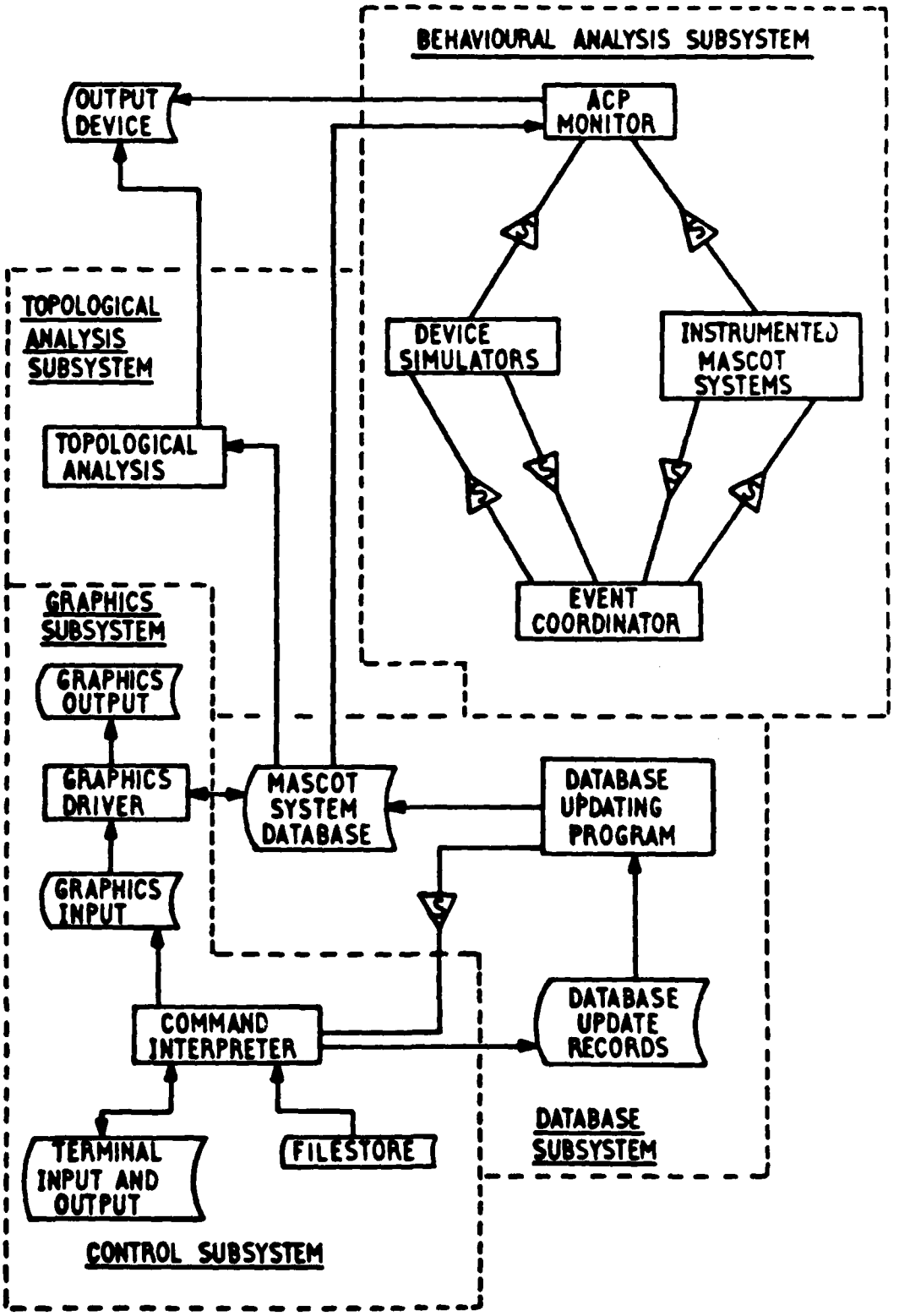

BEHAVIOURAL ANALYSIS SUBSYSTEM
OUTPUT DEVICE
ACP MONITOR
TOPOLOGICAL ANALYSIS SUBSYSTEM
DEVICE SIMULATORS
INSTRUMENTED MASCOT SYSTEMS
TOPOLOGICAL ANALYSIS
EVENT COORDINATOR
GRAPHICS SUBSYSTEM
GRAPHICS OUTPUT
GRAPHICS DRIVER
MASCOT SYSTEM DATABASE
DATABASE UPDATING PROGRAM
GRAPHICS INPUT
COMMAND INTERPRETER
DATABASE UPDATE RECORDS
DATABASE SUBSYSTEM
TERMINAL INPUT AND OUTPUT
FILESTORE
CONTROL SUBSYSTEM

FIG. 3·1

5 THE EVENT COORDINATOR

5.1 Introduction

Figure 3.1 shows that the behavioural analysis subsystem contains an event coordinator and chapter 3.2 introduced the notion of tasks. In this chapter the aim is to describe how the event coordinator task functions and to detail a set of facilities that it provides. These facilities enable the MASCOT ACP network to be simulated by a number of simulating tasks in conjunction with the event coordinator task. Chapters 7 and 8 describe how this set of facilities is used to simulate devices and MASCOT systems respectively.

5.2 Description

The simulating tasks generate a set of discrete future events. Each event has an associated simulation time at which we say the event occurs. At any point in simulated time a finite set of future events exists (one event per simulating task). The function of the event coordinator is to sort the times of the events and send a stimulus to each task that predicted the event in the order of event occurrence in simulated time. The stimulus informs the simulating task that the event has occurred. Each simulating task runs by predicting a single future event and awaiting the stimulus from the event coordinator before generating its next future event.

The event coordinator and each simulating task that is either a simulation of a MASCOT system (on a single processor) or of a device, run under an operating system that supports pseudo parallelism between the tasks. Simulating tasks communicate with the event coordinator by sending messages (simulation calls) to it and awaiting a reply. The message handling function is provided by the supporting operating system.

The event coordinator contains a clock that ticks in simulated time. Time now is the current time indicated by the clock. The simulation runs so that each simulating task is allowed initially to generate a single future event. Thus a period of simulated clock ticks from time now until the predicted time of the future event is created during which the task is simulating either a processor executing instructions or a device in operation. For a task simulating a MASCOT system this implies that a MASCOT activity is running uninterruptedly on the processor or that part of the MASCOT machine is running likewise. For a device simulator the period might represent the time taken to perform a data transfer with the outside world. The generation of such a future event is achieved by a simulation call named PROCESS FOR which takes a parameter specifying the time that is to elapse before the event occurs.

Once each simulating task has generated a future event it is awaiting a reply (the stimulus) from the event coordinator. Because all the simulating tasks will be waiting in this manner it is certain that no other event in simulated time can occur between time now and the time of the earliest future event which has been predicted. The event coordinator therefore updates the clock so that simulated time now becomes the time of the earliest future event. In this way a list of known future events is generated which indicates for each event the time of the event and the task which predicted it. This list is referred to as the event schedule.

Once simulated time has been updated a reply is sent to the task that generated the earliest future event and that event is removed from the event schedule. The event coordinator then waits until the same simulating task either predicts its next event or terminates itself. The event schedule is rechecked and the simulation continued in this way until the simulation time limit is reached.

So far we have assumed that the time at which the reply is sent by the event coordinator to the PROCESS FOR call is the time when the call was made plus the duration specified by the parameter of the call. However, in order to simulate the effect of devices interrupting the processor on which a MASCOT activity is executing we must allow events (representing the device interrupts) to occur during the PROCESS FOR period. There are two aspects to the provision of this facility. The first is how the task simulating a device generates an interrupt request and the second is how to inform the task simulating the MASCOT system that the interrupt has occurred.

A simulation call named ACTIVATE is provided in order to simulate an interrupt. It specifies the identity of another simulating task and indicates that the calling task is generating an interrupt of that task. The event coordinator sends no reply to this message. An event is generated at time now for the task which is to be ACTIVATEd (interrupted). The event coordinator awaits another call from the task that called ACTIVATE in order to predict the next future event for that task.

We now have the possibility that the event schedule contains more than one future event for a task. The ACTIVATEd task will have an event at time now and also one that was generated by its last call of PROCESS FOR. We say that the event at time now has preempted the task's next event and we have a chain of events for that task.

In order to describe how preemption of events is controlled we say that a task can consist of a number of threads. Without the concept of preemption each simulating task would be a single thread. Because an interrupt can temporarily suspend the execution of such a thread (called the base-level thread) and cause another sequence of instructions within the same task to be obeyed, the task now has two threads, only one of which can execute at a time. In theory a task can have one base-level thread and any number of other threads caused by nested preemption. The base-level thread exists until the task calls END TASK. All other threads are destroyed when each uses the simulation call named END PREEMPTION.

The chain of events for a given task then consists of a number of records each corresponding to a single thread with the base-level thread record at the back end of the chain. Each record requires to hold the time of the next event for the task as predicted by the thread and the time at which the thread was preempted (if it was). It is now clear that the END PREEMPTION simulation call needs to modify (delay) the predicted event time for the thread at the head of the chain, using time now and the time at which the thread was preempted to calculate the delay.

Figure 5.1 illustrates the operation of a two-level preemption. There is a base-level thread (1) and two preemption threads (2 and 3). The threads have predicted durations of p1, p2 and p3 respectively. Time is shown to progress along the page from left to right. The execution of

each thread is shown by horizontal lines and distinct threads are separated vertically. The vertical lines directed upwards indicate preemption of the thread below in favour of the thread above. Return to the preempted thread is indicated by the vertical lines directed downwards. Where a thread execution is shown by dotted lines this indicates the predicted point in time for the end of thread (marked by the word "predicted") and means that the thread was suspended during the period. The actual point in time at which the thread completes is marked by the word "actual".

The event coordinator on scanning the event schedule will find the task with the earliest event which might well be the task just ACTIVATEd. A reply is sent to that task which indicates that a preemption has occurred. It is then the responsibility of the preempted thread to deal with the situation. The preempted thread's end of PROCESS FOR event time will now require to be delayed until the preemption is complete. End of event preemption is indicated by the preempting thread using the simulation call END PREEMPTION. On receipt of this message the event coordinator delays the event time for the event at the head of the event chain for that task by the duration of the preemption and continues to scan the event schedule. No further reply is sent to the task until the event time is reached (unless another preemption occurs).

In addition to predicting a future event by means of the PROCESS FOR call a simulating task can specify that its next event will occur at the time when a given condition becomes true (or set) or after a length of time referred to as the timeout. In order to specify this a simulation call named AWAIT is provided which takes a condition number and timeout value as parameters. The event coordinator monitors the states of a number of condition flags each of which is set by the simulation call named SET CONDITION. As soon as the condition becomes set a reply is sent to any simulating tasks which have called AWAIT and have not timed out.

6 THE ACP MONITOR

6.1 Introduction

Paragraph 3.2 describes the calculations to be performed in order to measure performance of an ACP network design. To record the information required to complete these calculations requires that data be transmitted from instrumented MASCOT systems and device simulators to a data-gathering task. This is achieved by placing functions in the simulating tasks representing the system which send messages/data to the data gathering task called the ACP monitor.

6.2 Functional Description

The function of the ACP monitor is to act on data sent from a task that is either simulating a MASCOT machine or a device. Data is received in coded form and is converted by the monitor to a form suitable for human presentation. The monitor accesses the MASCOT system database (see chapter 9) to convert ACP node numbers into system element names as given on the ACP diagram. The monitor executes in a cyclic fashion reading messages from a FIFO queue and processing them in the order read. The necessary processing is as follows:-

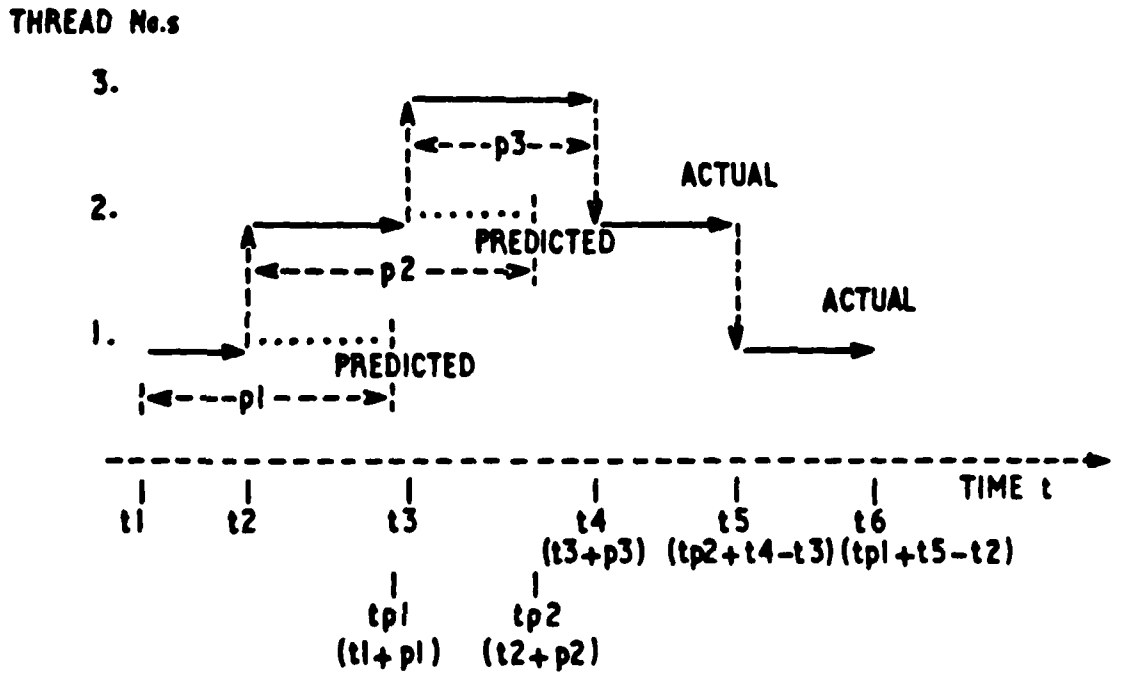

THREAD No.s
3.
2.
1.
p3
p2
p1
ACTUAL
PREDICTED
ACTUAL
PREDICTED
TIME t
t1
t2
t3
t4
(t3+p3)
t5
(tp2+t4-t3)
t6
(tp1+t5-t2)
tp1
(t1+p1)
tp2
(t2+p2)

FIG. 5·1

1 Read a message

2 Decode the message

3 Perform computations

4 Output results

6.3 Input Message Examples

Figures 6.1, 6.2 and 6.3 give an indication of the content of input messages received by the ACP monitor. They show message types for nodes of type activity, IDA and device respectively.

6.4 Data Computations

An initial set of computations is proposed:-

a amount of processor time used by each activity

b amount of processor time used by all activities on a given processor

c total waiting time by each activity on each control queue

d total waiting time by each activity on any control queue

e total time spent by system waiting on control queues

f total pending time for each activity on each control queue

g total pending time for each activity on any control queue

h total time spent by system pending on control queues

i total time for which an IDA buffer is full

j total time for which an IDA buffer is empty

k longest continuous period when an IDA buffer is full

l longest continuous period when an IDA buffer is empty

m total amount of data lost from each input device

n total amount of data lost to each output device

o average frequency at which data is read from each input device

p average frequency at which data is written to each output device.

6.5 Delivery of Results

When the event coordinator is started a simulation time limit is specified. During the time that the simulation is running the ACP monitor continuously records data and performs computations. When the simulation time limit is reached the performance results are available for output.

Type	Message Content
1	The amount of processor time consumed so far or since last message and the current value of simulation time now.
2	The amount of time spent by the activity waiting on a particular control queue so far or since last message. The simulation time at the start and end of the waiting period together with the control queue reference number.
3	The amount of time spent by the activity pending access to a particular control queue so far or since last message. The simulation time at the start and end of the pending period together with the control queue reference number.

Figure 6.1

Type	Message Content
1	The time that the IDA buffer becomes full
2	The time that the IDA buffer ceases to be full.
3	The time that the IDA buffer becomes empty.
4	The time that the IDA buffer ceases to be empty.

Figure 6.2

Type	Message Content
1	Indicates data lost on input
2	Indicates data lost on output
3	Indicates data read from an input device
4	Indicates data written to an output device.

Figure 6.3

7 DEVICE SIMULATING TASKS

7.1 Introduction

Device simulating tasks (also called device simulators) simulate data flow to/from hardware devices. Simulating tasks can model two types of device:-

i those which generate an interrupt to indicate either that data is ready for input or that an output data transfer has completed (or is requested); this type would also include interrupts from clock devices.

ii those which set a status register to indicate both that input data is available and that the device is free for the next output transfer.

Node attributes indicate whether the device is an input or output device and specify the rate at which input data is generated and output data transferred. The type of device (as given above, ie i or ii) is defined by the basic functional cycle.

7.2 The Basic Functional Cycle

Each device simulating task runs in an endless loop performing a series of calls on the event coordinator using the facilities described in chapter 5. As an example figure 7.1 illustrates the basic cycle for the two types of input device: non-interrupting and interrupting.

1 Non-interrupting Device

```
PROCESS FOR (device start up time)
reply := SET CONDITION ( c, time between inputs )
(c is the condition number)
if reply indicates "condition ignored" then report to the ACP
monitor.
```

2 Interrupting Device

```
PROCESS FOR( device start up time )
ACTIVATE( interrupt responding task number )
reply := AWAIT( data read condition, time before next interrupt)
IF reply indicates condition not set THEN report to ACP monitor
that input data is lost.
```

Figure 7.1

8 INSTRUMENTED MASCOT SYSTEMS

8.1 Introduction

Both the MASCOT machine and the ACP network that it supports are instrumented by calls to the event coordinator and the ACP monitor. These calls are planted in the code of the MASCOT machine and within activities, access procedures and interrupt handlers of the application ACP network. The following paragraphs illustrate by example how this might be done for a given MASCOT system.

8.2 The MASCOT Machine

The time taken to execute the various parts of the MASCOT machine will be gauged by planting calls to the ACP monitor in the code as the example in figure 8.1 for execution of the JOIN primitive shows.

It is clear from this figure that access to the source code of the MASCOT machine is needed. This may not always be possible and is one reason for considering MASCOT machine characterisation and interpretation as described in chapter 10.

8.3 MASCOT Modules

MASCOT modules are instrumented in a similar manner to the MASCOT machine. Note that preemption by interrupt must be explicitly checked for after a PROCESS FOR call. Figures 8.2, 8.3 and 8.4 illustrate instrumentation of activities, access procedures and interrupt handlers respectively.

```
send to ACP monitor(time now)
enter JOIN code
PROCESS FOR (join entry period)
check control queue
PROCESS FOR (check time)
alter control queue fields
PROCESS FOR (...)
send to ACP monitor (time now)
return to caller (PROCESS FOR ...)
or reschedule (PROCESS FOR ...)
```

Figure 8.1

```
send to ACP monitor (time now)
reply := PROCESS FOR (period 1)
check reply and interrupt
send to ACP monitor (time now)
call access procedure
send to ACP monitor (time now)
reply := PROCESS FOR (period 2)
check reply and interrupt
etc....
```

Figure 8.2 - activities

```
send to ACP monitor (time now)
send to ACP monitor (JOIN called)
JOIN control queue
send to ACP monitor (time now)
send to ACP monitor (at head of queue)
check IDA state
reply := PROCESS FOR (check time)
check reply and interrupt
send to ACP monitor (time now)
WAIT control queue
send to ACP monitor (time now)
STIM control queue
LEAVE control queue
send to ACP monitor (LEAVE called)
etc....
```

Figure 8.3 - access procedures

```
send to ACP monitor (time now)
read device data
SET CONDITION (data read)
STIMINT control queue
send to ACP monitor (time now)
END HANDLER (END PREEMPTION)
```

Figure 8.4 - interrupt handlers

9 THE MASCOT SYSTEM DATABASE DESIGN

9.1 The ACP Representation

For representation in the computer the ACP network is considered to be a directed graph (in the formal graph-theoretic sense). The nodes of the directed graph correspond to the ACP nodes. The arcs of the directed graph match the oriented connections between nodes. For the purpose of analysis we constrain the structure of the ACP network such that it maps onto a connected directed graph. This means that starting from any node in the network and ignoring the direction of data flow indicated by node connections, it is possible to trace a path to all other nodes in the network.

The connectivity of the graph is given by the adjacency matrix. This is a square matrix of order n, the total number of nodes in the graph. The elements [aij] of the adjacency matrix A are given by:-

a[ij] = label on arc from node i to node j, if the arc exists
and
zero otherwise.

9.2 Node Attributes

Three categories of node attribute are identified:-

i Topological or pictorial. These are placement details derived via the graphics input medium (ie from the user).

ii MASCOT module. These will include node type, node name, execution profiles (see chapter 10), IDA structure etc.. The information is derived from user input.

iii Environment. These include details of the hardware with which each node is associated, eg for activities details of the processor on which they execute and the MASCOT machine with which the node is associated.

9.3 Attribute Management

Attributes need to be created, deleted, modified, listed and attached to and detached from nodes. To support this requirement the MASCOT System Database contains two attribute tables. The first table is called the Attribute-Indexed Table which, as its name implies, is indexed by attribute serial number. Each attribute has an entry in the table that points to the data structure for that attribute. The table supports the functions of attribute creation, deletion, modification and listing.

The second table is called the Node-Indexed Attribute Table and is used to attach attributes to nodes. This table supports the functions of attaching and detaching node attributes.

It is assumed that any attempt to access the details of an attribute is accompanied by knowledge of the data structure for that attribute. It is quite possible that two nodes will require to have the same attribute, eg IDAs may have identical structures. In this case there is no need to duplicate the attribute data structure, but simply to arrange that the table pointers refer to the appropriate data structure.

9.4 The Database Updating Program

This program performs the functions of attribute management described in paragraph 9.3. It reads a set of records from a file produced by the control subsystem called database update records and writes to the MASCOT System Database accordingly. For update records specifying a listing request, the requested data is sent to the command interpreter.

10 CHARACTERISATION OF SIMULATING TASKS

10.1 Introduction

In order to model the behaviour of the system represented by the ACP network and its associated devices a number of simulating tasks are created for the system.

In chapters 7 and 8 it is assumed that the simulating tasks exist as programs within which it is possible to plant calls both to the event coordinator and the ACP monitor. In theory then, any MASCOT machine that has been designed can be instrumented in this way. However, in practice it might be that access to the source code of a particular MASCOT machine is denied or that a chosen MASCOT machine cannot readily interface to the simulated environment. To overcome this problem simulating tasks are created from a set of descriptions written in a non-implementation-dependent language.

A task simulating a MASCOT machine is built from a generalised MASCOT machine description, a MASCOT machine profile and a MASCOT interpreter that executes according to these descriptions. The generalised MASCOT machine description is an attempt to provide the essential functional features needed to represent the operation of a MASCOT machine. This description remains static. For each different design of MASCOT machine a MASCOT machine profile exists detailing MASCOT primitive time overheads and scheduling policy.

The MASCOT interpreter simulates a MASCOT machine and controls activities using a set of instances. Instances for each activity and IDA are created by means of an instance creation program. This program reads ACP node attributes from the MASCOT system database and generates instances.

The ACP node attributes are generated from a set of profiles written in a description language.

A task simulating a device is built using information held in a device profile contained in the MASCOT system database and the task is created by the instance creation program. These tasks are called device simulators.

Figure 10.1 illustrates the above and shows where the MASCOT interpreter appears in the behavioural subsystem. Note that in a typical system there will be more than one MASCOT interpreter and several device simulators.

10.2 The Generalised MASCOT Machine

The generalised MASCOT machine constitutes the set of functions necessary to operate the MASCOT primitives. These functions are realised by a set of procedures that are instrumented with calls to the ACP monitor and the event coordinator. There are nine procedures that support the basic primitives JOIN, WAIT, LEAVE, STIM, SUSPEND and DELAY and control the scheduling of activities. Figure 10.2 shows how these procedures interrelate.

To illustrate how the MASCOT machine functions are supported figures 10.3 and 10.4 show the procedures for ENDSLICE and SCHEDULER respectively.

10.3 MASCOT Machine Profiles

A MASCOT machine is characterised by its primitive procedures, its scheduling policy and its interrupt strategy. All the information required to describe these features is held in a MASCOT machine profile. The profile is used by the generalised MASCOT machine to determine the operation of scheduling and interrupt handling and in deciding the PROCESS FOR periods to be set up in conjunction with the event coordinator. Figure 10.5 shows the information that the profile contains.

10.4 ACP Node Profiles

10.4.1 Activities

Activities are described by a sequence of execution periods which will be some combination of:

a access procedure calls

and

b local processing statements.

In addition, a reference to the processor on which the activity runs is given.

An activity execution profile contains:

- name

- processing sequence = ordered list of processing steps.

The possible steps are:

Type 1	local processing	- a figure specifying the number of instructions
Type 2	access call	- reference to an IDA and an access profile within it

- number of times sequence is to be obeyed

10.4.2 IDAs

Channels are described by:

a data buffer size

b access protocol (eg wait or poll if buffer empty/full)

Pools are described by an access protocol which defines a period of mutual exclusion and access procedure times.

An IDA profile contains:

- name

- data buffer size (in words)

- message length (channels only). A message is defined as that entity which is passed into/out of a channel by a single call of the access procedure.

- array of references to access profiles.

Each profile contains the following:

- access identifier (name or number)

- access call entry overhead (number of instructions)

- access call exit overhead (number of instructions)

- buffer-check overhead (number of instructions)

- data transfer overhead (number of instructions)

- buffer state-change overhead (number of instructions)

- access type = reader, writer or readwriter

- access execution profile = ordered list of accessing steps.
  The possible step types are shown below:

  Type 1 JOIN

  Type 2 check buffer state;
  IF full/empty THEN WAIT;
  transfer data, amend buffer state

  Type 3 check buffer state;
  IF full/empty THEN no action
  ELSE transfer data, amend buffer state

  Type 4 STIM

  Type 5 LEAVE

  Type 6 transfer pool data

  Type 7 SUSPEND

  Type 8 DELAY

10.4.3 Devices

A device profile contains:

- name

- unit of data transfer (in bytes)

- operating sequence = ordered list of operating steps

The possible steps are:

  Type 1 WAIT FOR COMMAND

  Type 2 SET STATUS(n)
  n=1 device free
  n=2 device busy
  n=4 data available
  n=8 data not available

  Type 3 TRANSFER DATA

  Type 4 INTERRUPT

  Type 5 PAUSE - time delay

- number of times sequence is to be obeyed

10.5 Language Description of Profiles

In order to generate the profiles that are to be held in the MASCOT system database as node attributes, a language is defined by the syntax shown in Figure 10.6.

10.6 Instance Creation

10.6.1 Introduction

Instances are of three types: activity, IDA and device and are created using an instance creation program. The program creates and initialises instances from profiles and connectivity information held in the MASCOT system database. A profile is an example of a node attribute as described in chapter 9.

10.6.2 Instance Initialisation

Instances are initialised using attribute data from the MASCOT system database.

10.7 The MASCOT Interpreter

The MASCOT interpreter interprets the generalised description of a MASCOT machine in conjunction with its MASCOT machine profile and a set of activity and IDA instances in order to produce an instrumented MASCOT system that runs in the behavioural analysis subsystem. It runs as a sequential program, assuming that only a single processor is available on which to run its activities.

The program executes by interpreting the content of activity instances in conjunction with IDA instances, the generalised MASCOT machine description and a MASCOT machine profile.

therefore updates the clock so that simulated time now becomes the time of the earliest future event. In this way a list of known future events is generated which indicates for each event the time of the event and the task which predicted it. This list is referred to as the event schedule.

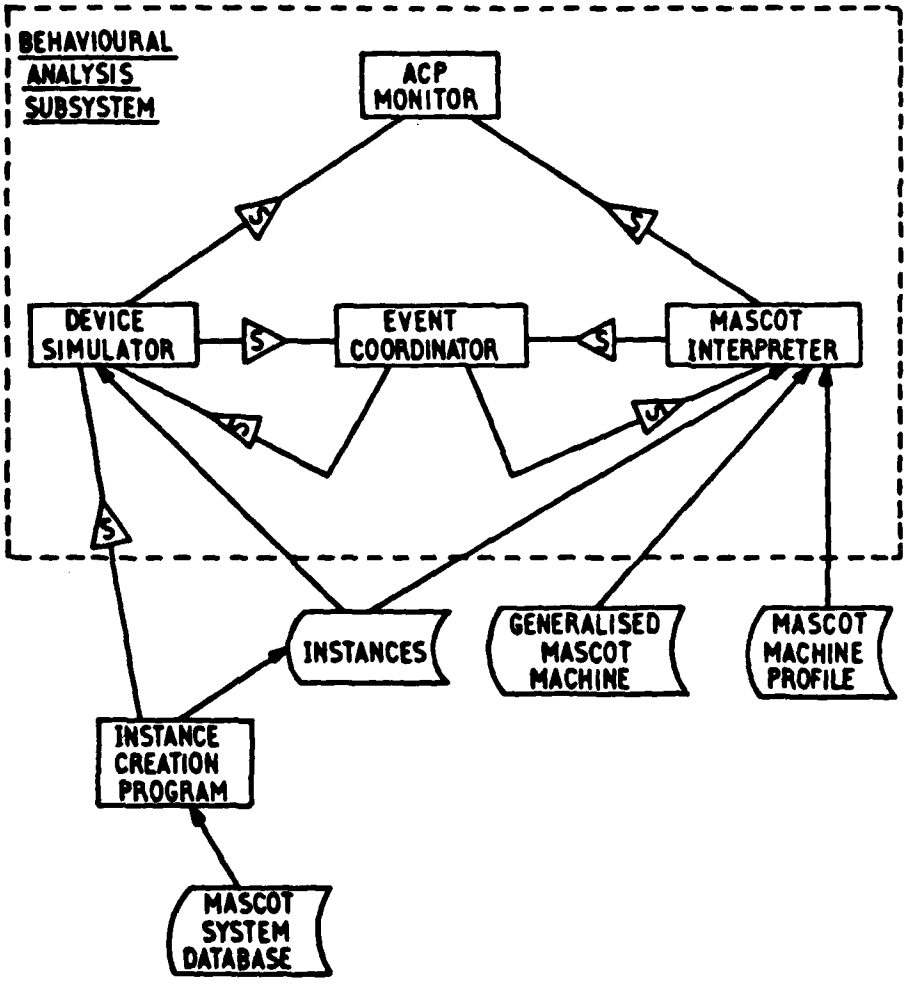


FIG.10·1

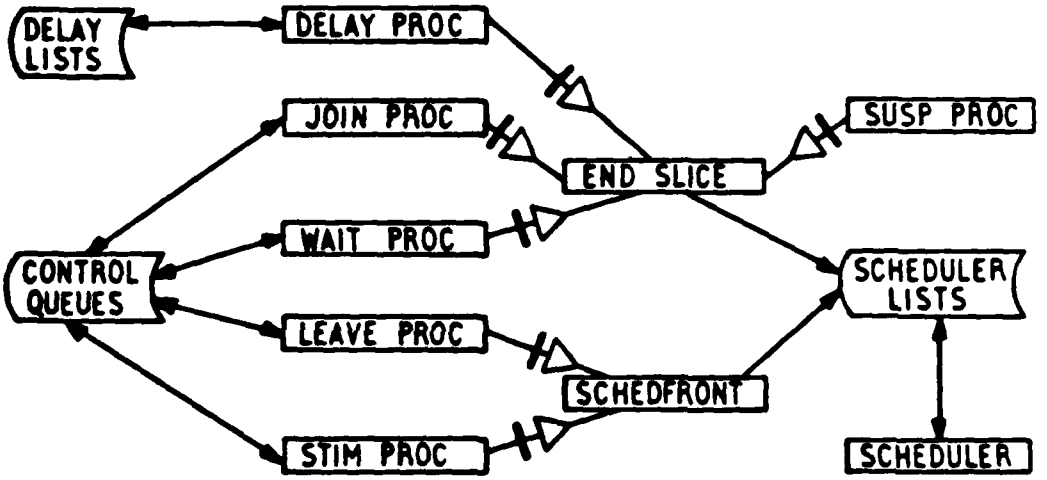


FIG. 10·2

ENDSLICE

This procedure is called by JOIN PROC and WAIT PROC.

Inputs: A reference to an activity,
reference to a control queue,
indication of whether the activity is to be
flagged pending or waiting,

Function: PROCESS FOR (ENDSLICE overhead * instruction speed
of processor type specified in
current activity instance).
Add the activity to the pending list of the
referenced control queue
- at the front if it is waiting or at the back if
pending. An entry in the list includes:

- the referenced activity's identity
- whether activity is pending or waiting
- time now (ie entry time).

Call SCHEDULER.

Figure 10.3

SCHEDULER

This procedure is called by END SLICE.

Inputs: a list of runnable activities. An entry in the list contains:-

- activity identity

- priority of activity

Function: Read MASCOT machine profile to determine scheduling algorithm and make scheduling decision.
PROCESS FOR (scheduling decision time).
Send slice-start time to ACP monitor.
Enter the scheduled activity.

Figure 10.4

- JOIN overhead if queue free

- JOIN overhead if queue not free (does not include pending time)

- WAIT overhead if queue already stimmed

- WAIT overhead if no stim received (does not include waiting time)

- STIM overhead

- LEAVE overhead

- SUSPEND overhead - time taken to add the activity to the scheduler list

- SCHEDFRONT overhead - time taken to transfer to the scheduler list an activity released by a call of LEAVE

- DELAY overhead - time taken to add an activity to and remove it from a delay list

- ENDSLICE overhead - time taken to add an activity to a pending list

- STIM reschedule policy - does a reschedule take place?

- LEAVE reschedule policy - does a reschedule take place?

- scheduling rule for selecting an activity to run from a list of runnable activities

Figure 10.5

```
1   ACTIVITY <activity type name> =
      (<list of IDA types>):
        (sequence of calls of :-
          PROCESSFOR : <integer number of instructions>
          and ACCESS : <ida type> : <access name>
      ) * <replicator>


2   DEVICE <device type name> =
      (<IDA type>) : ... then as for ACTIVITY


3   IDA <IDA type name> =
      (calls of :-
        BUFFSIZE : <integer units>
       and
        MESSLENGTH : <integer units> ):
      (calls of :- ACCESS <access name>
        (<READER,WRITER or READWRITER>) =
          (<sequence of the following calls>:-
            ENTRY : <integer number of instructions>
            EXIT : <integer number of instructions>
            CHECK : <integer number of instructions>
            XFER : <integer number of instructions>
            CHANGESTATE : <integer number of instructions>
          (<sequence of the following :-
            JOIN
            WAITCHECK or POLLCHECK
            STIM
            LEAVE
            POOL
            SUSPEND
            DELAY >);


4   PROCESSOR <integer type> =
      (SPEED: <integer basic units>,
       MASCOT MACHINE: <name>)


5   MASCOT MACHINE <type name> =
      (JOIN(FREE) : <integer basic units>,
       JOIN(NOTFREE) : <integer basic units>,
       WAIT(STIMMED): <integer basic units>,
       WAIT(NOTSTIMMED): <integer basic units>,
       STIM: <integer basic units>,
       LEAVE: <integer basic units>,
       SUSPEND: <integer basic units>,
       DELAY: <integer basic units>,
       SCHEDFRONT: <integer basic units>,
       ENDSLICE: <integer basic units>,
       SCHEDULING: STIM POLICY = <abstract description>,
       LEAVE POLICY = <abstract description>,
       SELECTION RULE = <abstract description> )
```

Figure 10.6

11 THE CONTROL SUBSYSTEM DESIGN

11.1 Introduction

The command interpreter reads input mainly from a keyboard but can, on keyborad command, switch its input source to a filestore. It generates input to the graphics subsystem, produces database update records and can output database information to the terminal.

11.2 Graphics Commands

All keyboard input is checked for commands that relate to manipulation of the graphics terminal output. These commands are sent to the graphics subsystem without alteration.

11.3 Analysis Control Commands

These commands specify that analysis of the ACP network is to start and what type of analysis is to be done (topological or behavioural). The command interpreter responds by activating the relevant programs.

11.4 Database Update Records

The database update records are created as input to the database program. There are three types of database update record that the command interpreter produces. Each is produced by the input of one of the following types of command:-

1 Construction Commands

By means of a command syntax the structure of the ACP network can be described. The update records contain the node connectivity data.

2 Attribute Commands

The attribute commands and their function are shown in figure 11.1. Command iv expects the database updating program to return data from the MASCOT system database which the command interpreter can print on the terminal or send to the filestore.

3 Profile Description Statements

These statements describe the profiles for activities, devices, IDAs, processors and MASCOT machines.

	Command	Function
i	ADD ATTRIBUTE	Creation of new attributes
ii	REMOVE ATTRIBUTE	Deletion of attributes
iii	UPDATE ATTRIBUTE	Modification of attribute data structures
iv	LIST ATTRIBUTES	Listing of attributes
v	ATTACH ATTRIBUTE	Attaching of attributes to nodes
vi	DETACH ATTRIBUTE	Detaching of attributes from nodes

Figure 11.1

12 SUMMARY

This paper describes a tool to facilitate analysis of MASCOT systems. Both topological and behavioural analysis of an ACP network can be performed using techniques of graph theory and event simulation. The simulation environment comprises a number of simulating tasks running under a general purpose operating system that supports pseudo parallelism between the tasks. A database is created to hold the ACP diagram including node attributes. The topological input is derived from either a set of MASCOT construction commands or from a graphics terminal. Node attributes used in the simulation are created from a set of language statements describing in skeleton form the nature of MASCOT activities, IDAs and devices. Performance data is recorded using a data monitoring program.

The paper proposes that existing MASCOT machines be instrumented to interface to the simulation program and the data monitor. For use where this is not possible (and to make the tool more general purpose) a method of characterising MASCOT machines from language statements and interpreting these characterisations is presented.

13 DEVELOPMENT STATUS OF THE TOOL

The tool has been designed to the pre-implementation stage, with the exception of the graphics subsystem and the generalised characterisation technique described in chapter 10. Implementation of an experimental tool on the Ferranti Argus 700, running under the OSC56 operating system has begun. This tool comprises the event coordinator and ACP monitor programs. The facilities offered by these programs will be used to instrument the Argus Hosted Frozen MASCOT System (reference [2]). Following the successful conclusion to this work the MASCOT System Database and the associated utilities will be implemented and integrated with the tool. Subsequently the topological analysis and graphics subsystems will be implemented.

Detailed design and implementation of the generalised characterisation technique is considered to involve a significant effort and is likely to follow the first production of the tool as a separate development.

BIBLIOGRAPHY

1 Carre, B., "Graphs and Networks", University of Southampton 1979.

2 Tyler, A. C. F., "A Hosted Frozen MASCOT Machine for the Ferranti Argus 700", RSRE Report No 81001, October 1981.

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMORANDUM 3475		UNCLASSIFIED

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
	ROYAL SIGNALS AND RADAR ESTABLISHMENT, MALVERN
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title
A TOOL FOR ANALYSING THE DESIGN AND PERFORMANCE OF A MASCOT ACP NETWORK - INTRODUCTORY REPORT

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
TYLER, A C F				

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
C & GR GROUP

Descriptors (or keywords)

MASCOT
ACP NETWORK

continue on separate piece of paper

Abstract This paper describes a tool to facilitate analysis of MASCOT systems. Both topological and behavioural analysis of an ACP network can be performed using techniques of graph theory and event simulation. The simulation environment comprises a number of simulating tasks running under a general purpose operating system that supports pseudo parallelism between the tasks. A database is created to hold the ACP diagram including node attributes. The topological input is derived from either a set of MASCOT construction commands or from a graphics terminal. Node attributes used in the simulation are created from a set of language statements describing in skeleton form the nature of MASCOT activities, IDAs and devices. Performance data is recorded using a data monitoring program.

(contd)

The paper proposes that existing MASCOT machines be instrumented to interface to the simulation program and the data monitor. For use where this is not possible (and to make the tool more general purpose) a method of characterising MASCOT machines from language statements and interpreting these characterisations is presented.

END

FILMED